```
# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR
# NOTEBOOK.
import kagglehub
arishmam_sales_data_path = kagglehub.dataset_download('arishmam/sales-data')

print('Data source import complete.')
```

```
import pandas as pd
df = pd.read_csv('/kaggle/input/sales-data/kaggle sale.csv')
```

Preprocessing

```
print(df.head())
```

```
    User ID  Gender  Age  EstimatedSalary  Purchased satisfied
0  15624510    Male   19            19000          0         no
1  15810944    Male   35            20000          0         no
2  15668575  Female   26            43000          0         no
3  15603246  Female   27            57000          0         no
4  15804002    Male   19            76000          0         no
```

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64
 3   EstimatedSalary  400 non-null    int64
 4   Purchased        400 non-null    int64
 5   satisfied        400 non-null    object
dtypes: int64(4), object(2)
memory usage: 18.9+ KB
None
```

```
df['satisfied '].head()
```

```
0    no
1    no
2    no
3    no
4    no
Name: satisfied , dtype: object
```

```
df = df.rename(columns = {'satisfied ':'satisfied'})
print(df.head())
```

```
    User ID  Gender  Age  EstimatedSalary  Purchased satisfied
0  15624510    Male   19            19000          0        no
1  15810944    Male   35            20000          0        no
2  15668575  Female   26            43000          0        no
3  15603246  Female   27            57000          0        no
4  15804002    Male   19            76000          0        no
```

```
df = df.drop(columns=['User ID'])
```

```
print(df.head())
```

```
    Gender  Age  EstimatedSalary  Purchased satisfied
0    Male   19            19000          0        no
1    Male   35            20000          0        no
2  Female   26            43000          0        no
3  Female   27            57000          0        no
4    Male   19            76000          0        no
```

```
from sklearn.preprocessing import LabelEncoder


le = LabelEncoder()


columns = ['Gender', 'satisfied']
for i in columns:
    df[i] = le.fit_transform(df[i])

# columns = ['Gender', 'satisfied']  # Ensure the column names match the modified names

# for col in columns:
#     if col in df.columns:  # Check if the column exists
#         df[col] = le.fit_transform(df[col])
#     else:
#         print(f"Column '{col}' not found in DataFrame.")


print(df.head())
```

```
   Gender  Age  EstimatedSalary  Purchased  satisfied
0       1   19            19000          0          0
1       1   35            20000          0          0
2       0   26            43000          0          0
3       0   27            57000          0          0
4       1   19            76000          0          0
```

```
x = df.iloc[:,:-1]
print(x.head())
```

```
   Gender  Age  EstimatedSalary  Purchased
0       1   19            19000          0
1       1   35            20000          0
2       0   26            43000          0
3       0   27            57000          0
4       1   19            76000          0
```

```
y = df.iloc[:,-1]
print(y.head())
```

```
0    0
1    0
2    0
3    0
4    0
Name: satisfied, dtype: int64
```

```
print(x.shape)
print(y.shape)
```

```
(400, 4)
(400,)
```

```
from sklearn.model_selection import train_test_split


x_train,x_test,y_train,y_test = train_test_split(x,y,random_state=42)


print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(x_test.shape)
```

```
(300, 4)
(100, 4)
(300,)
(100, 4)
```

DecisionTree

```
from sklearn.tree import DecisionTreeClassifier
```

```
dtc = DecisionTreeClassifier(criterion='gini',
                             splitter='best',
                             random_state=42)


dtc.fit(x_train,y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```

```
prediction = dtc.predict(x_test)


from sklearn.metrics import accuracy_score,classification_report,confusion_matrix


print("Decision Tree Classifier Accuracy:", accuracy_score(y_test, prediction))
```

```
Decision Tree Classifier Accuracy: 0.69
```

```
print("Classification Report:\n", classification_report(y_test, prediction))
```

```
Classification Report:
                   precision    recall  f1-score   support

               0       0.50      0.39      0.44        31
               1       0.75      0.83      0.79        69

        accuracy                           0.69       100
       macro avg       0.62      0.61      0.61       100
    weighted avg       0.67      0.69      0.68       100
```

```
print("Confusion Matrix:\n", confusion_matrix(y_test, prediction))
```

```
Confusion Matrix:
 [[12 19]
 [12 57]]
```

Naive Base Classifier

```
from sklearn.naive_bayes import GaussianNB


gnc = GaussianNB()


gnc.fit(x_train,y_train)
```

```
▾ GaussianNB
GaussianNB()
```

```
pred = gnc.predict(x_test)


print("Naive Bayes Classifier Accuracy:", accuracy_score(y_test, pred))
```

```
Naive Bayes Classifier Accuracy: 0.75
```

```
print("Classification Report:\n", classification_report(y_test, pred))
```

```
Classification Report:
                   precision    recall  f1-score   support

               0       0.69      0.35      0.47        31
               1       0.76      0.93      0.84        69

        accuracy                           0.75       100
       macro avg       0.72      0.64      0.65       100
    weighted avg       0.74      0.75      0.72       100
```

```
print("Confusion Matrix:\n", confusion_matrix(y_test, pred))
```

```
Confusion Matrix:
 [[11 20]
 [ 5 64]]
```